Inclass - Lab (Week 1)

About the data set 1 (Life Expectancy data)

The dataset contains information about several health and economic factors that contribute to average life expectancy for different countries.
Attribute information:

Country: Name of the country

Status: Whether the country is Developed or Developing

Adult_Mortality: Mortality rate for age group 15-60 out of every 1000 individuals of the population

Infant_Deaths: Number of infant deaths per 1000 population

Hepatitis: Hepatitis B (HepB) immunization coverage for 1 year olds (Percentage)

Measles: Number of reported cases for measles per 1000 from population

BMI: Average Body Mass Index for entire population

Underfive_Deaths: Number of deaths under 5 years of age per 1000 population

Polio: Polio (Pol3) immunization coverage for 1 year olds (Percentage)

Diphtheria: Diphtheria tetanus toxoid and pertussis (DTP3) immunization coverage for 1 year olds (Percentage)

HIV: Deaths per 1000 live births due to HIV/AIDS (0-4 years)

GDP: Gross Domestic Product per capita (in USD)

Population: Population of the country

Malnourished10_19: Prevalence of malnutrition among children and adolescents for Age 10 to 19 (Percentage)

Malnourished5_9: Prevalence of malnutrition among children for Age 5 to 9 (Percentage)

Income_Index: Human Development Index (HDI) in terms of national income per capita (index ranging from 0 to 1)

Schooling: Number of years of Schooling

Life_Expectancy: Life Expectancy in age for the country

Table of Content

Import the required libraries

In [254]:
```
# type your code here
#used to perform dataframe related operations
import pandas as pd
#user to perform any mathematical operations
import numpy as np
#visualization
import seaborn as sns
import matplotlib.pyplot as plt
#for scaling
from sklearn.preprocessing import MinMaxScaler,StandardScaler,RobustScaler #use std scaler only when the data is norma
#for transformation
from sklearn.preprocessing import PowerTransformer
#warnings
from warnings import filterwarnings
filterwarnings('ignore')

from sklearn.model_selection import train_test_split
#for performing linear regression
from statsmodels.api import OLS
from sklearn.linear_model import LinearRegression

#for testing performance of model
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_absolute_percentage_error
from sklearn.metrics import mean_squared_error

#for multicollinearity treatment
from statsmodels.stats.outliers_influence import variance_inflation_factor

9#for testing normality of residuals
from statsmodels.graphics.gofplots import qqplot

from statsmodels.api import add_constant
```

Load the first dataset and check the first five observations

Load the csv file and set the first column as index

In [177]:
```
# load the csv file
data = pd.read_csv('Life_Expectancy.csv')
# type your code here
data.head()
```

Out[177]:

	Country	Status	Adult_Mortality	Infant_Deaths	Hepatitis	Measles	BMI	Underfive_Deaths	Polio	Diphtheria	HIV	GDP	Popula
0	Afghanistan	Developing	263	62	65	1154	19.1	83	6	65	0.1	584.259210	33736
1	Albania	Developing	8	0	98	0	57.2	1	98	98	0.1	4575.763787	288
2	Algeria	Developing	19	21	95	63	59.5	24	95	95	0.1	4132.762920	39871
3	Angola	Developing	335	66	64	118	23.3	98	7	64	1.9	3695.793748	2785
4	Argentina	Developing	116	8	94	0	62.8	9	93	94	0.1	13467.123600	43417

In [178]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 124 entries, 0 to 123
Data columns (total 18 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Country             124 non-null    object
 1   Status              124 non-null    object
 2   Adult_Mortality     124 non-null    int64
 3   Infant_Deaths       124 non-null    int64
 4   Hepatitis           124 non-null    int64
 5   Measles             124 non-null    int64
 6    BMI                124 non-null    float64
 7   Underfive_Deaths    124 non-null    int64
 8   Polio               124 non-null    int64
 9   Diphtheria          124 non-null    int64
 10   HIV                124 non-null    float64
 11  GDP                 124 non-null    float64
 12  Population          124 non-null    int64
 13  Malnourished10_19   124 non-null    float64
```

In [179]:
```
data.describe()
```

Out[179]:

	Adult_Mortality	Infant_Deaths	Hepatitis	Measles	BMI	Underfive_Deaths	Polio	Diphtheria	HIV	GDP
count	124.000000	124.000000	124.000000	124.000000	124.000000	124.000000	124.000000	124.000000	124.000000	124.000000
mean	160.475806	27.637097	85.104839	2841.637097	41.733871	36.798387	86.637097	87.919355	0.819355	6866.816502
std	101.794506	94.347510	20.839229	11517.586950	21.186385	122.248912	21.653342	19.122879	1.764127	10885.243579
min	3.000000	0.000000	6.000000	0.000000	2.100000	0.000000	6.000000	6.000000	0.100000	5.668726
25%	73.500000	1.000000	82.000000	0.750000	20.950000	1.000000	88.000000	89.000000	0.100000	639.799727
50%	139.500000	10.000000	94.000000	37.000000	54.100000	12.000000	95.000000	95.000000	0.100000	2728.291765
75%	234.000000	21.000000	97.000000	588.750000	59.750000	26.500000	98.000000	98.000000	0.400000	8437.562893
max	438.000000	910.000000	99.000000	90387.000000	71.400000	1100.000000	99.000000	99.000000	8.100000	56554.387600

In [180]:
```
data.describe(include= object)
```

Out[180]:

	Country	Status
count	124	124
unique	124	2
top	Afghanistan	Developing
freq	1	105

Let's begin with some hands-on practice exercises

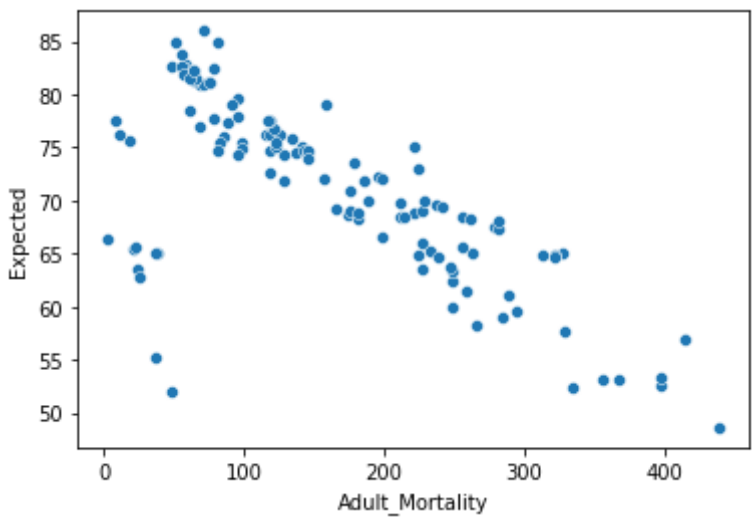
1. Bivariate Regression

We shall use the life expectancy dataset

1. How do we analyze the impact of adult mortality rate on average life expectancy of a country?

In [181]:
```
sns.scatterplot(data['Adult_Mortality'],data['Expected'])
```

Out[181]:
```
<AxesSubplot:xlabel='Adult_Mortality', ylabel='Expected'>
```



In [182]:
```
data['Adult_Mortality'].corr(data['Expected'])
```

Out[182]:
```
-0.7108678534954708
```

In [183]:
```
data[['Adult_Mortality' ,'Expected']].corr()
```

Out[183]:

	Adult_Mortality	Expected
Adult_Mortality	1.000000	-0.710868
Expected	-0.710868	1.000000

2. A regression model is built to check the impact of Human Development Index (Income) on the life expectancy of a nation. What is the expected error value?

In [184]:
```
X = add_constant(data['Income_Index'])
y = data['Expected']
single_model = OLS(y,X).fit()
single_model.summary()
```

Out[184]:

OLS Regression Results

Dep. Variable:	Expected	**R-squared:**	0.711
Model:	OLS	**Adj. R-squared:**	0.709
Method:	Least Squares	**F-statistic:**	300.0
Date:	Mon, 06 Mar 2023	**Prob (F-statistic):**	1.13e-34
Time:	20:09:38	**Log-Likelihood:**	-363.40
No. Observations:	124	**AIC:**	730.8
Df Residuals:	122	**BIC:**	736.4
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	41.9191	1.741	24.081	0.000	38.473	45.365
Income_Index	42.4243	2.449	17.322	0.000	37.576	47.273

Omnibus:	11.870	**Durbin-Watson:**	1.355
Prob(Omnibus):	0.003	**Jarque-Bera (JB):**	26.475
Skew:	0.280	**Prob(JB):**	1.78e-06
Kurtosis:	5.193	**Cond. No.**	8.87

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [185]:
```
ypred_single_model=single_model.predict(X)
```

In [186]:
```
mean_squared_error(y,ypred_single_model,squared=True)
```

Out[186]:
```
20.56116776979112
```

In [187]:
```
np.sum((y-ypred_single_model)**2)/y.shape[0]
```

Out[187]:
```
20.56116776979112
```

2. Multiple Linear Regression(MLR)

3. Analyze the impact of immunization on life expectancy.

In [188]:
```
X = add_constant(data[['Hepatitis ','Polio','Diphtheria ']])
y = data['Expected']
multi_model = OLS(y,X).fit()
multi_model.summary()
```

Out[188]:

OLS Regression Results

Dep. Variable:	Expected	**R-squared:**	0.138
Model:	OLS	**Adj. R-squared:**	0.116
Method:	Least Squares	**F-statistic:**	6.395
Date:	Mon, 06 Mar 2023	**Prob (F-statistic):**	0.000468
Time:	20:09:38	**Log-Likelihood:**	-431.15
No. Observations:	124	**AIC:**	870.3
Df Residuals:	120	**BIC:**	881.6
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	57.9881	3.524	16.453	0.000	51.010	64.966

In [189]:
```
#life_expectancy=(57.9881)inetercept-(0.1037)hepatitis+(0.0910)polio+(0.1613)diptheria+error
```

4. If the information that whether the country is a developed or a developing country is included, does it alter the impact of immunzation on life expectancy ?

In [190]:
```
data['Status'].unique()
```

Out[190]:
```
array(['Developing', 'Developed'], dtype=object)
```

In [191]:
```
data['Status'] = data['Status'].map({'Developing': 0 , 'Developed' : 1})
```

In [192]:
```
data['Status'].value_counts(normalize = True)
```

Out[192]:
```
0    0.846774
1    0.153226
Name: Status, dtype: float64
```

In [193]:
```
X = pd.concat([X,data['Status']],axis = 1)
```

In [194]:
```
#X = X.drop(['Status'],axis=1)
```

In [ ]:
```

```

In [195]:
```
mlr_model_status = OLS(y,X).fit()
mlr_model_status.summary()
```

Out[195]:

OLS Regression Results

Dep. Variable:	Expected	**R-squared:**	0.321
Model:	OLS	**Adj. R-squared:**	0.298
Method:	Least Squares	**F-statistic:**	14.06
Date:	Mon, 06 Mar 2023	**Prob (F-statistic):**	2.00e-09
Time:	20:09:38	**Log-Likelihood:**	-416.35
No. Observations:	124	**AIC:**	842.7
Df Residuals:	119	**BIC:**	856.8
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	59.2054	3.148	18.805	0.000	52.971	65.439
Hepatitis	-0.0035	0.063	-0.056	0.955	-0.128	0.121
Polio	0.0771	0.039	1.993	0.049	0.000	0.154
Diphtheria	0.0456	0.076	0.602	0.548	-0.104	0.196
Status	10.6036	1.872	5.664	0.000	6.897	14.310

Omnibus:	2.973	**Durbin-Watson:**	1.211
Prob(Omnibus):	0.226	**Jarque-Bera (JB):**	2.635
Skew:	-0.355	**Prob(JB):**	0.268
Kurtosis:	3.077	**Cond. No.**	758.

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

3. Inferences about slope

5. Find the significant variables in the full model when all the variables are considered for prediction of life expectancy.

In [196]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
data['Country']=le.fit_transform(data['Country'])
```

In [197]:
```
X = add_constant(data.drop(columns =['Country','Expected']))
y = data['Expected']
full_model = OLS(y,X).fit()
full_model.summary()
```

Out[197]:

OLS Regression Results

Dep. Variable:	Expected	**R-squared:**	0.848
Model:	OLS	**Adj. R-squared:**	0.826
Method:	Least Squares	**F-statistic:**	37.43
Date:	Mon, 06 Mar 2023	**Prob (F-statistic):**	1.94e-36
Time:	20:09:38	**Log-Likelihood:**	-323.38
No. Observations:	124	**AIC:**	680.8
Df Residuals:	107	**BIC:**	728.7
Df Model:	16		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	58.0318	3.284	17.673	0.000	51.522	64.541

In [198]:
```
X.drop(columns = ['Polio'],inplace  = True)
```

In [199]:
```
full_model = OLS(y,X).fit()
full_model.summary()
```

Out[199]:

OLS Regression Results

Dep. Variable:	Expected	**R-squared:**	0.848
Model:	OLS	**Adj. R-squared:**	0.827
Method:	Least Squares	**F-statistic:**	40.22
Date:	Mon, 06 Mar 2023	**Prob (F-statistic):**	3.27e-37
Time:	20:09:38	**Log-Likelihood:**	-323.48
No. Observations:	124	**AIC:**	679.0
Df Residuals:	108	**BIC:**	724.1
Df Model:	15		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	57.8644	3.247	17.823	0.000	51.429	64.300
Status	2.0827	1.158	1.798	0.075	-0.213	4.379
Adult_Mortality	-0.0229	0.004	-5.693	0.000	-0.031	-0.015
Infant_Deaths	0.0529	0.036	1.462	0.147	-0.019	0.125
Hepatitis	0.0656	0.032	2.064	0.041	0.003	0.128
Measles	-5.728e-05	6.05e-05	-0.947	0.346	-0.000	6.26e-05
BMI	-0.0280	0.022	-1.260	0.210	-0.072	0.016
Underfive_Deaths	-0.0371	0.027	-1.395	0.166	-0.090	0.016
Diphtheria	-0.0411	0.036	-1.126	0.263	-0.113	0.031
HIV	-1.0591	0.262	-4.042	0.000	-1.578	-0.540
GDP	5.098e-05	3.46e-05	1.475	0.143	-1.76e-05	0.000
Population	-1.749e-08	8.25e-09	-2.120	0.036	-3.38e-08	-1.14e-09
Malnourished10_19	-0.4554	0.291	-1.567	0.120	-1.031	0.121
Malnourished5_9	0.2796	0.279	1.004	0.318	-0.272	0.832
Income_Index	18.4344	6.267	2.942	0.004	6.012	30.857
Schooling	0.3657	0.344	1.062	0.291	-0.317	1.048

Omnibus:	5.844	**Durbin-Watson:**	1.747
Prob(Omnibus):	0.054	**Jarque-Bera (JB):**	8.480
Skew:	0.143	**Prob(JB):**	0.0144
Kurtosis:	4.249	**Cond. No.**	9.12e+08

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 9.12e+08. This might indicate that there are
strong multicollinearity or other numerical problems.

In [200]:
```
X.columns
```

Out[200]:
```
Index(['const', 'Status', 'Adult_Mortality', 'Infant_Deaths', 'Hepatitis ',
       'Measles ', ' BMI ', 'Underfive_Deaths ', 'Diphtheria ', ' HIV', 'GDP',
       'Population', 'Malnourished10_19 ', 'Malnourished5_9', 'Income_Index',
       'Schooling'],
      dtype='object')
```

In [201]:
```
X = add_constant(X[['Status','Adult_Mortality','Hepatitis ',' HIV','Population','Income_Index']])
y = data['Expected']
signi_model = OLS(y,X).fit()
signi_model.summary()
```

Out[201]:

OLS Regression Results

Dep. Variable:	Expected	**R-squared:**	0.834
Model:	OLS	**Adj. R-squared:**	0.825
Method:	Least Squares	**F-statistic:**	97.84
Date:	Mon, 06 Mar 2023	**Prob (F-statistic):**	3.20e-43
Time:	20:09:38	**Log-Likelihood:**	-329.08
No. Observations:	124	**AIC:**	672.2
Df Residuals:	117	**BIC:**	691.9
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	56.2304	2.552	22.030	0.000	51.176	61.285
Status	2.6723	1.077	2.481	0.015	0.539	4.806
Adult_Mortality	-0.0235	0.004	-5.950	0.000	-0.031	-0.016
Hepatitis	0.0353	0.016	2.237	0.027	0.004	0.067
HIV	-1.0942	0.252	-4.342	0.000	-1.593	-0.595
Population	-1.698e-08	8.1e-09	-2.096	0.038	-3.3e-08	-9.36e-10
Income_Index	24.0067	3.117	7.701	0.000	17.833	30.181

Omnibus:	10.733	**Durbin-Watson:**	1.751
Prob(Omnibus):	0.005	**Jarque-Bera (JB):**	21.171
Skew:	0.288	**Prob(JB):**	2.53e-05
Kurtosis:	4.941	**Cond. No.**	5.50e+08

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 5.5e+08. This might indicate that there are
strong multicollinearity or other numerical problems.

6. If economic indices are used to predict the life expectancy, calculate the total variation for the observed values of life expectancy.

In [202]:
```
X= add_constant(data[['GDP','Income_Index']])
model_eco = OLS(y,X).fit()
model_eco.summary()
```

Out[202]:

OLS Regression Results

Dep. Variable:	Expected	**R-squared:**	0.713
Model:	OLS	**Adj. R-squared:**	0.709
Method:	Least Squares	**F-statistic:**	150.6
Date:	Mon, 06 Mar 2023	**Prob (F-statistic):**	1.44e-33
Time:	20:09:38	**Log-Likelihood:**	-362.85
No. Observations:	124	**AIC:**	731.7
Df Residuals:	121	**BIC:**	740.2
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	42.6132	1.865	22.848	0.000	38.921	46.306
GDP	4.515e-05	4.36e-05	1.034	0.303	-4.13e-05	0.000
Income_Index	40.9705	2.823	14.512	0.000	35.381	46.560

Omnibus:	9.157	**Durbin-Watson:**	1.349
Prob(Omnibus):	0.010	**Jarque-Bera (JB):**	16.954
Skew:	0.232	**Prob(JB):**	0.000208
Kurtosis:	4.751	**Cond. No.**	1.05e+05

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.05e+05. This might indicate that there are
strong multicollinearity or other numerical problems.

In [203]:
```
ypred_eco = model_eco.predict(X)
```

In [204]:
```
mean_squared_error(y,ypred_eco,squared=False)
```

Out[204]:
```
4.514523051891257
```

7. Determine the lower bound and upper bound for estimated value of life expectancy per unit change in HDI (for income)

In [205]:
```
X = add_constant(X['Income_Index'])
model_idx = OLS(y,X).fit()
ypred_idx = model_idx.predict(X)
mean_squared_error(y,ypred_idx,squared=False)
```

Out[205]:
```
4.534442387966918
```

In [206]:
```
print(ypred_idx.min())
print(ypred_idx.max())
```

```
56.003993287261274
82.01007873026995
```

4. Model Evaluation

8. A model is built to analyze the impact of mortality on Life expectancy. Would the addition of information about population of the country lead to a better prediction?

In [207]:
```
data.columns
```

Out[207]:
```
Index(['Country', 'Status', 'Adult_Mortality', 'Infant_Deaths', 'Hepatitis ',
       'Measles ', ' BMI ', 'Underfive_Deaths ', 'Polio', 'Diphtheria ',
       ' HIV', 'GDP', 'Population', 'Malnourished10_19 ', 'Malnourished5_9',
       'Income_Index', 'Schooling', 'Expected'],
      dtype='object')
```

In [208]:
```
X = add_constant(data[['Adult_Mortality','Infant_Deaths','Underfive_Deaths ']])
y = data['Expected']
model_mort = OLS(y,X).fit()
ypred_model_mort = model_mort.predict(X)
mean_squared_error(y,ypred_model_mort,squared=False)
```

Out[208]: 5.779226662185357

In [209]:
```
X = add_constant(data[['Adult_Mortality','Infant_Deaths','Underfive_Deaths ','Population']])
y = data['Expected']
model_mort_pop = OLS(y,X).fit()
ypred_mort_pop = model_mort_pop.predict(X)
mean_squared_error(y,ypred_mort_pop,squared=False)
```

Out[209]: 5.776058728648563

In [210]:
```
model_mort_pop.summary()
```

Out[210]:

OLS Regression Results

Dep. Variable:	Expected	**R-squared:**	0.531
Model:	OLS	**Adj. R-squared:**	0.515
Method:	Least Squares	**F-statistic:**	33.68
Date:	Mon, 06 Mar 2023	**Prob (F-statistic):**	8.93e-19
Time:	20:09:39	**Log-Likelihood:**	-393.41
No. Observations:	124	**AIC:**	796.8
Df Residuals:	119	**BIC:**	810.9
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	80.3071	1.019	78.813	0.000	78.289	82.325
Adult_Mortality	-0.0550	0.006	-9.929	0.000	-0.066	-0.044
Infant_Deaths	0.0886	0.050	1.788	0.076	-0.010	0.187
Underfive_Deaths	-0.0760	0.038	-1.975	0.051	-0.152	0.000
Population	4.736e-09	1.31e-08	0.361	0.718	-2.12e-08	3.07e-08

Omnibus:	54.418	**Durbin-Watson:**	1.537
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	143.042
Skew:	-1.738	**Prob(JB):**	8.69e-32
Kurtosis:	6.949	**Cond. No.**	8.72e+07

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 8.72e+07. This might indicate that there are
strong multicollinearity or other numerical problems.

9. Fit a full model and test the significance of the overall model.

In [211]:
```
# type your code here
#In Q3.5 we had fitted full model aand the p(fstats) indicates the model is siginificant
```

4. Assumptions of Linear Regression

10. A model is built using all the features (full model) to predict the life expectancy. Check whether the residuals obtained form the model follow a normal distribution.

In [212]:
```
# type your code here
#In Q3.5 we had fitted full model and p(jb), p(omnibus) values indicate that the residules are not normally distribute
```

11. Does the interaction of status of the country and its GDP have significant effect on life expectancy? (consider the full model)

In [213]:
```
data['interact'] = data['GDP'] * data['Status']
X = add_constant(data.drop(columns = ['Country','Expected']))
full_model=OLS(y,X).fit()
full_model.summary()
```

Out[213]:

OLS Regression Results

Dep. Variable:	Expected	**R-squared:**	0.853
Model:	OLS	**Adj. R-squared:**	0.830
Method:	Least Squares	**F-statistic:**	36.30
Date:	Mon, 06 Mar 2023	**Prob (F-statistic):**	2.14e-36
Time:	20:09:39	**Log-Likelihood:**	-321.30
No. Observations:	124	**AIC:**	678.6
Df Residuals:	106	**BIC:**	729.4
Df Model:	17		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	58.7216	3.265	17.988	0.000	52.249	65.194

In [225]:
```
X = X.drop('Diphtheria ',axis = 1)
full_model = OLS(y,X).fit()
full_model.summary()
```

Out[225]:

OLS Regression Results

Dep. Variable:	Expected	**R-squared:**	0.840
Model:	OLS	**Adj. R-squared:**	0.829
Method:	Least Squares	**F-statistic:**	75.71
Date:	Mon, 06 Mar 2023	**Prob (F-statistic):**	3.11e-42
Time:	20:13:53	**Log-Likelihood:**	-326.56
No. Observations:	124	**AIC:**	671.1
Df Residuals:	115	**BIC:**	696.5
Df Model:	8		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	57.7284	2.617	22.061	0.000	52.545	62.912

In [ ]:
```
# there is no effect on interaction variable in the predection
```

12. Construct a full no-intecept model. What would you conclude when all the variables take value 0?

In [ ]:
```
# type your code here
```

Hereon on we shall consider the FEV dataset.

About the data set (Respiratory function data)

Dataset consists of information about respiratory function in children and adolescents and factors that might impact the respiratory function.
Attribute information:

Age: Age in years

Height: Height in inches

Gender: Gender value is 1 if male 0 if female

Smoke: Whether the person is a smoker or non- smoker, the value is 1 if smoker and 0 otherwise

FEV: Forced Exhalation Volume (FEV), a measure of how much air somebody can forcibly exhale from their lung (Percentage)

Load the dataset and check the first five observations

In [227]:
```
# load the csv file
# type your code here
data1 = pd.read_csv('LungCapdata.csv')
data1
```

Out[227]:

	Age	Height	Gender	Smoke	FEV
0	9	57.0	0	0	1.708
1	8	67.5	0	0	1.724
2	7	54.5	0	0	1.720
3	9	53.0	1	0	1.558
4	9	57.0	1	0	1.895
...	...	...	...	...	...
649	16	67.0	1	1	4.270
650	15	68.0	1	1	3.727
651	18	60.0	0	0	2.853
652	16	63.0	0	1	2.795
653	15	66.5	0	0	3.211

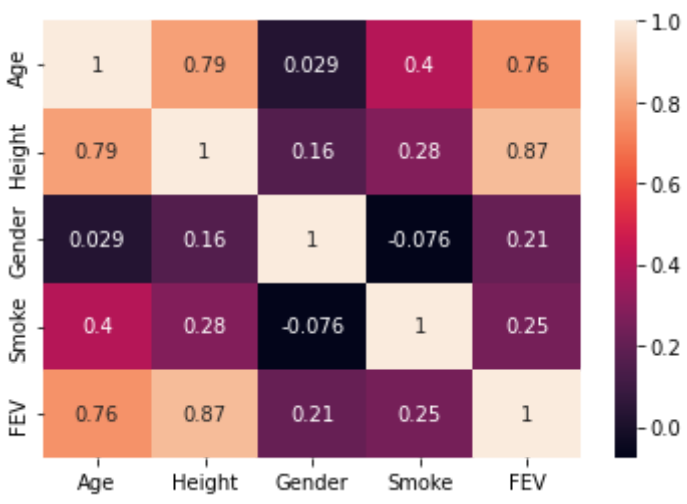
654 rows × 5 columns

In [229]:
```
data1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 654 entries, 0 to 653
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Age     654 non-null    int64  
 1   Height  654 non-null    float64
 2   Gender  654 non-null    int64  
 3   Smoke   654 non-null    int64  
 4   FEV     654 non-null    float64
dtypes: float64(2), int64(3)
memory usage: 25.7 KB
```

In [231]:
```
sns.heatmap(data1.corr(),annot = True)
```

Out[231]: <AxesSubplot:>



13. Can we use a linear regression model to analyze the impact of all the features on respiratory function (FEV) ?

In [232]:
```
X = add_constant(data1.drop(['FEV'],axis = 1))
y = data1['FEV']
model = OLS(y,X).fit()
model.summary()
```

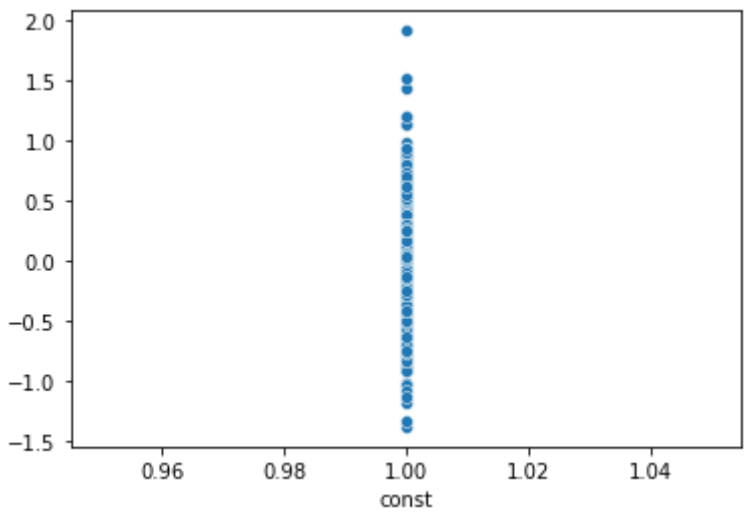
Out[232]:

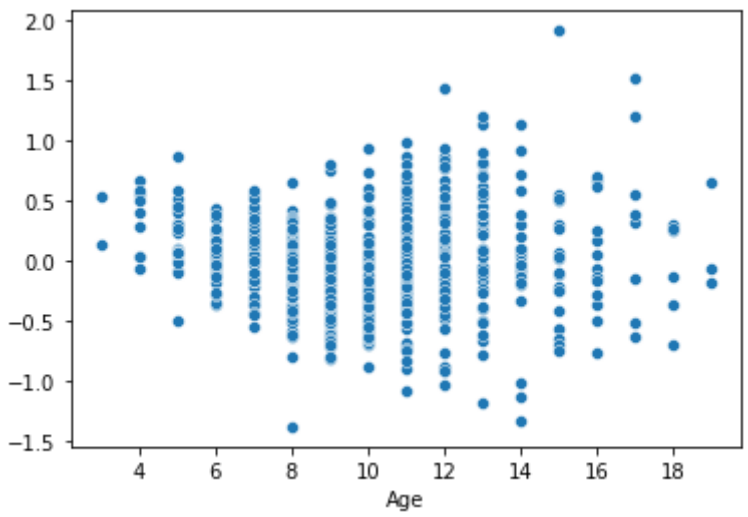
OLS Regression Results

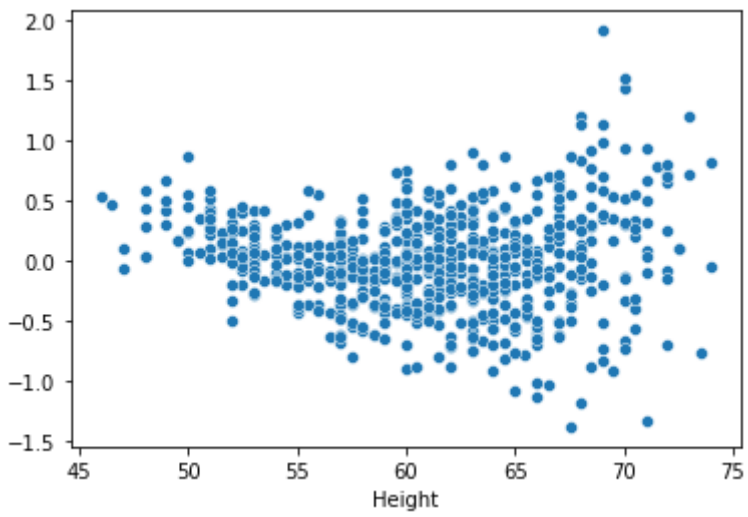
Dep. Variable:	FEV	**R-squared:**	0.775
Model:	OLS	**Adj. R-squared:**	0.774
Method:	Least Squares	**F-statistic:**	560.0
Date:	Mon, 06 Mar 2023	**Prob (F-statistic):**	9.10e-209
Time:	20:19:44	**Log-Likelihood:**	-345.90
No. Observations:	654	**AIC:**	701.8
Df Residuals:	649	**BIC:**	724.2
Df Model:	4		
Covariance Type:	nonrobust		

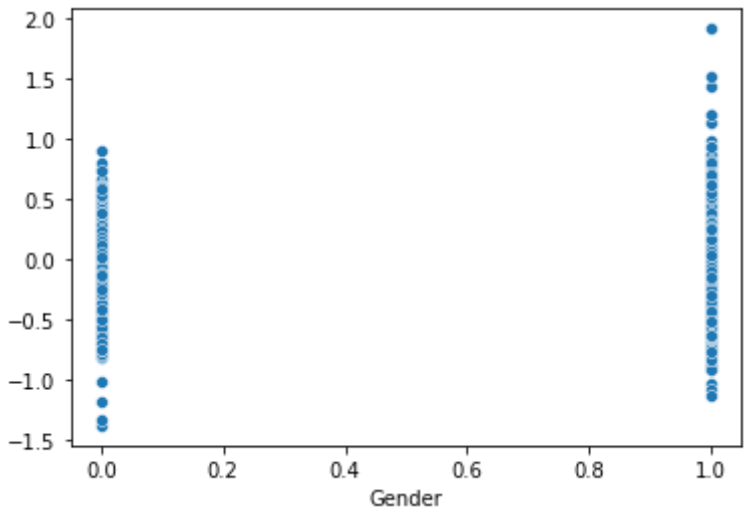
	coef	std err	t	P>|t|	[0.025	0.975]
const	-4.4570	0.223	-20.001	0.000	-4.895	-4.019

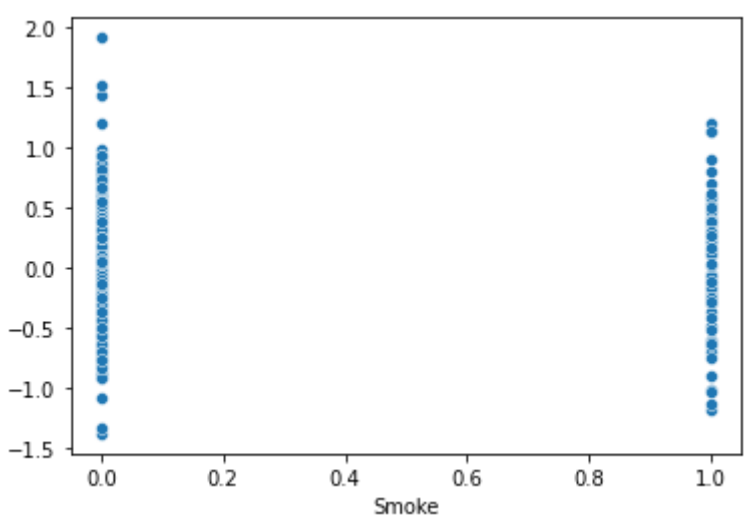
In [233]:
```
residuals = model.resid

for i in X.columns:
    sns.scatterplot(X[i],residuals)
    plt.show()
```











14. How can we rectify the multicollinearity detected in question 13?

In [234]:
```
#type your code
#vif = 1/(1-0.9) = 10
#vif = 1/(1-0.8) = 5
#y = x1+x2+x3+x4
#x1 = x2+x3+x4 -> R2
```

In [237]:
```
X.drop(columns='const',inplace = True)
```

In [238]:
```
vif = pd.DataFrame()
vif['vif_factor'] = [variance_inflation_factor(X.values,i) for i in range(X.shape[1])]
```

In [239]:
```
vif['attributes'] = X.columns
```

In [240]:
```
vif
```

Out[240]:

	vif_factor	attributes
0	24.322098	Age
1	24.524524	Height
2	2.130851	Gender
3	1.342164	Smoke

In [242]:
```
X.drop(columns = 'Height',inplace = True)
```

In [243]:
```
vif = pd.DataFrame()
vif['vif_factor'] = [variance_inflation_factor(X.values,i) for i in range(X.shape[1])]
vif['attributes'] = X.columns
vif
```

Out[243]:

	vif_factor	attributes
0	2.307311	Age
1	1.977845	Gender
2	1.238003	Smoke

In [245]:
```
X = add_constant(X)
y = data1['FEV']
model = OLS(y,X).fit()
```

In [246]:
```
model.summary()
```

Out[246]:

OLS Regression Results

Dep. Variable:	FEV	**R-squared:**	0.609
Model:	OLS	**Adj. R-squared:**	0.608
Method:	Least Squares	**F-statistic:**	337.9
Date:	Mon, 06 Mar 2023	**Prob (F-statistic):**	3.45e-132
Time:	20:32:49	**Log-Likelihood:**	-526.84
No. Observations:	654	**AIC:**	1062.
Df Residuals:	650	**BIC:**	1080.
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.2378	0.080	2.964	0.003	0.080	0.395
Age	0.2268	0.008	28.765	0.000	0.211	0.242
Gender	0.3153	0.043	7.382	0.000	0.231	0.399
Smoke	-0.1540	0.078	-1.975	0.049	-0.307	-0.001

Omnibus:	18.151	**Durbin-Watson:**	1.597
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	20.607
Skew:	0.340	**Prob(JB):**	3.35e-05
Kurtosis:	3.542	**Cond. No.**	44.0

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

15. A Linear regression model is used to analyze the impact of all possible features on respiratory function (FEV). How do we check whether or not the model adequately captures the relationship between the response and predictor variables?

In [ ]:
```
#refer q13 for full model
#pval of fstats indicates model is significant
#pval of tstats suggest except the feature smoke rest of the feature have an impact on the prediction
#the model capture 60% of explained variation in the target and hence capture the corresponding relation between the r
```

16. A Linear regression model is used to analyze the impact of all possible features on respiratory function (FEV). Check whether the error terms are serially independent?

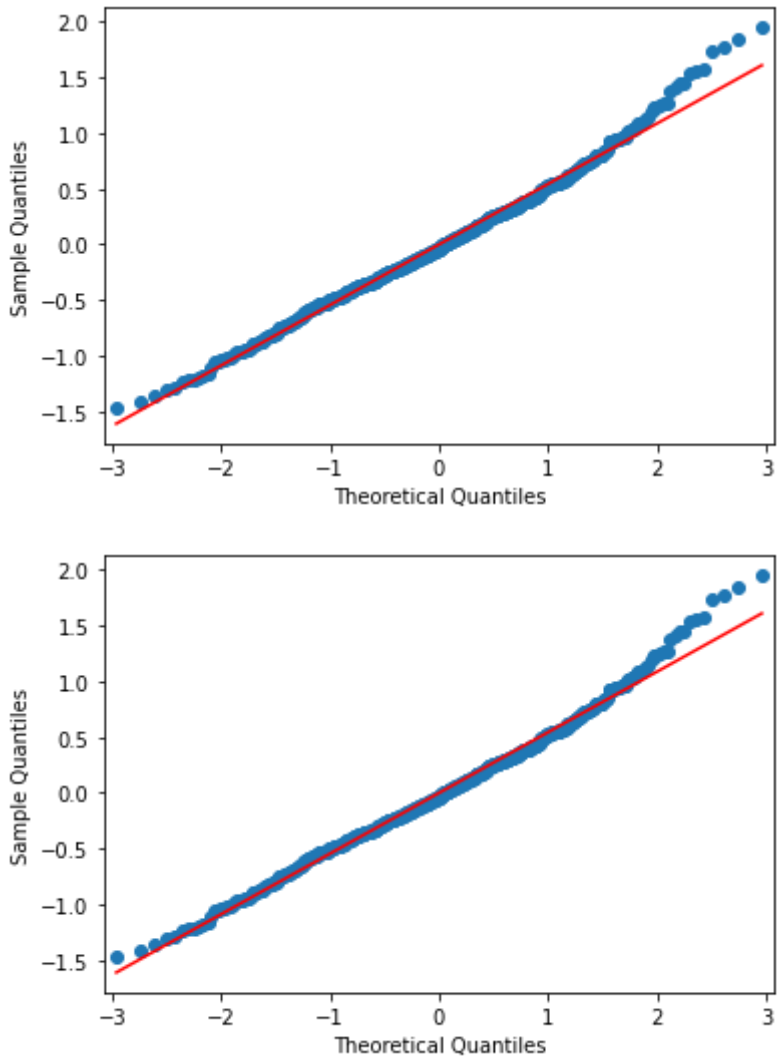
In [ ]:
```
#DW no. for full model (refer q13) indicates  no auto-correlation is present and the error terms are not serially depe
```

17. A model is built using all the features to predict the FEV. Check whether the residuals obtained form the model are normally distributed.

In [ ]:
```
#refer q13 for full model
# p(jb) < 0.05 indicating the residuals are not normally distributed
```

In [247]:
```
qqplot(model.resid, line = 'r')
```

Out[247]:



In [ ]:
```
#looking at the qq plot the poinys and the redline do not overlap indicating non-normal residuals
```

6. Model Performance

18. Consider the full model. Calculate the mean square error and the root mean square error.

In [248]:
```
X = add_constant(data1.drop(['FEV'],axis = 1))
y = data1['FEV']
model = OLS(y,X).fit()
model.summary()
```

Out[248]:

OLS Regression Results

Dep. Variable:	FEV	**R-squared:**	0.775
Model:	OLS	**Adj. R-squared:**	0.774
Method:	Least Squares	**F-statistic:**	560.0
Date:	Mon, 06 Mar 2023	**Prob (F-statistic):**	9.10e-209
Time:	22:47:27	**Log-Likelihood:**	-345.90
No. Observations:	654	**AIC:**	701.8
Df Residuals:	649	**BIC:**	724.2
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-4.4570	0.223	-20.001	0.000	-4.895	-4.019
Age	0.0655	0.009	6.904	0.000	0.047	0.084
Height	0.1042	0.005	21.901	0.000	0.095	0.114
Gender	0.1571	0.033	4.731	0.000	0.092	0.222
Smoke	-0.0872	0.059	-1.472	0.141	-0.204	0.029

Omnibus:	22.758	**Durbin-Watson:**	1.645
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	43.271
Skew:	0.207	**Prob(JB):**	4.02e-10
Kurtosis:	4.190	**Cond. No.**	861.

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [249]:
```
ypred = model.predict(X)
```

In [251]:
```
print('MSE',mean_squared_error(y,ypred))
print('rmse',mean_squared_error(y,ypred,squared = False))
```

```
MSE 0.16862317131301383
rmse 0.41063751815075766
```

19. Calculate the mean absolute error and the mean absolute percentage error. Compare the values of RMSE and MAE.

In [255]:
```
print('MSE',mean_absolute_error(y,ypred))
print('MAPE',mean_absolute_percentage_error(y,ypred))
```

```
MSE 0.31304644197549053
MAPE 0.1257087546714721
```

In [ ]:
```
#MAE<RMSE
#since both of them are error the model with less MAE and RMSE will be preffered
#m1->RMSE,MAE
#m2->RMSE,MAE
```

In [256]:
```
np.sum(np.abs(y-ypred)/y.shape[0])
```

Out[256]:
```
0.3130464419754905
```

20. Consider the full model, has the model overfitted?

In [257]:
```
xtrain,xtest,ytrain,ytest = train_test_split(X,y,random_state=1)
model = OLS(ytrain,xtrain).fit()
ypred_train = model.predict(xtrain)
ypred_test = model.predict(xtest)
```

In [259]:
```
print('Train score')
print('RMSE',mean_squared_error(ytrain,ypred_train,squared = False))
print('MAE',mean_absolute_error(ytrain,ypred_train))

print('Test score')
print('RMSE',mean_squared_error(ytest,ypred_test,squared = False))
print('MAE',mean_absolute_error(ytest,ypred_test))
```

```
Train score
RMSE 0.41285157457660115
MAE 0.31589481105775097
Test score
RMSE 0.4043718122440465
MAE 0.30530497391777905
```

In [261]:
```
xtrain.index
```

Out[261]:
```
Int64Index([622, 438, 259, 479,  40, 255,   5, 172, 386,  38,
            ...
            281, 390, 508, 583, 129, 144, 645,  72, 235,  37],
           dtype='int64', length=490)
```

In [262]:
```
#overfitting is a scenario where in the trian score is much better than test score
#thats not the case with current data hence we can conclude there is no over fitting
```

In [ ]:
```

```